NTCIP Mib's XML Storage Mechanism

Recording and Manipulating the NTCIP Management Information Base variables as XML using JDom and XPath

Edwards and Kelcey Technical Design Document for CHART2.

A study of the mechanism to load, store and manage the Mibs for an NTCIP compliant DMS using XML in CHART2. The NTCIP standard specifies a large number of Mib variables for the Signs to expose, query and modify. The most common manner of representing the Mibs is through the ASN.1 standard. The standard though nodular doesn't translate well into Object Oriented programming with it's current flat file structure. This makes the Mibs difficult to parse as a flat file, the alternative is to embed that information into the source code, but this doesn't separate the knowledge of the Mibs from the source. When representing the Mibs in source the depth of information would be name:value pair and Java type. This would also cause a proliferation of Mibs represented in source. An ideal solution will make the Mibs and their values, essentially transparent to the CHART2 System and only of interest to the DMSDeviceStatus.

Author	Cameron Riley
Email	criley@ekmail.com
Company	Edwards and Kelcey Technology, Inc
URL	http://www.ekcorp.com/
Address	750 Miller Drive, Suite F-1
City	Leesburg, Virginia
Zip	20175
phone	703.779.7988
fax	703.779.7989
date	5th November 2001

ABSTRACT

The Mib's as XML is a solution to the problem of representing, manipulating and querying Mib's in the NTCIP Message Signs. CHART2 as a distributed software system allows for the a single instance of the Mib structure to be stored on the network, this structure an be cloned by the DMS that is being manipulated and used locally on the client machine for querying and displaying the DMS status, as well as manipulating the DMS.

Representing Mib's as XML in CHART2 will solve the design issues in the NTCIP Driver project of, the large number of Mib's, an Object Oriented approach to representing the Mib's, Mib expansion and contraction, structure distribution and cloning across a distributed network, the non-java-friendly ASN.1 standard in flat file form, Mib nodularity and the large amount of information contained in each Mib.

CHART

CHART is the highway incident management program of the Maryland State Highway Administration. CHART is comprised of four major components, traffic monitoring, incident response, traveler information and traffic management. The traveler information component of CHART provides real-time information concerning travel conditions on main roads in the primary coverage area. The information is conveyed to travelers via either Variable Message Signs, Highway Advisory Radio, Commercial radio and television broadcasts and a telephone advisory service. The variable message signs are programmable message boards, both permanent and portable which are capable of displaying real-time traffic conditions to motorists. NTCIP compliant Dynamic Message Signs are a type of VMS. CHART2 is the current implementation of the CHART system being developed by the Maryland State Highways Administration.

The CHART2 System uses CORBA, to transport and communicate between objects across the distributed client and server system. The GUI's from which the DMS Devices are managed, exist as Clients in the CORBA system. The FMSServer carries the implementations of the Objects. The Driver interacts across this system by the IDL interfaces.

The ORB is a message bus between objects that may reside on any machine in the network. The IDL is the interfaces by which the distributed objects publish their capabilities. CORBA is the specification that describes the ORB's functionality. CORBA specifies location transparency and language transparency. For CHART2's distributed system, the former is the most important.

NTCIP

The National Transportation Communications for Intelligent Transportation System Protocol (NTCIP) is a family of standards maintained by NEMA, AASHTO and ITE. The NTCIP standards provide the rules and vocabulary for electronic traffic equipment from different manufacturers to communicate and operate with each other. NTCIP compliant devices must follow this standard. For a Device to be NTCIP compliant it must implement a mandatory set of MIB's, accept data transport by PMPP and be capable of reading Multi Message Format for sign display.

NTCIP Mib's

The NTCIP compliant DMS's support the TS-3.2, TS-3.3, TS-3.4, TS-3.5, TS-3.6 and TS-3.7 series of Mib's with some of the groups mandatory and others optional. The Mib descriptions are known as immutable through the NTCIP specification and hence can be loaded as objects following the Singleton Pattern into the CHART2 runtime. There are two areas' the Mib's need to be represented in the system, as an interface for the Mib tree including each Mib's attributes. The second area is as an instance for the specific DMS.

Mandatory Mib's under the NTCIP Specification;

- Configuration Conformance Group
- Security Node Conformance Group
- Sign Configuration and Capability Conformance Group

- Font Definition Conformance Group
- DMS Configuration Conformance Group
- Multiconfiguration Conformance Group
- Message Table Conformance Group
- Sign Control Conformance Group
- Illumination/Brightness Conformance Group
- Scheduling Conformance Group
- Sign Status Conformance Group

Mib Problem Domain

NTCIP poses many problems with Mib's and CHART2. The NTCIP standard specifies a large number of Mib's that the CHART2 NTCIP DMS Protocol Handler will need to be aware of, present, query and manipulate. This poses problems in the coding of the NTCIP Driver and requires an elegant solution to minimize code and time without compromising ease of code maintenance and flexibility.

Issue's surrounding NTCIP Mib's;

- The NTCIP Standard contains a large number of available Mib's both mandatory and optional. One manner of representing the Mib's in the system would be through hard coding them into objects as class member variables. However due to the large number of Mib's in the NTCIP specification this would make the code unwieldy, long, difficult to maintain and inflexible to changing requirements. This approach is also not an ideal Object Oriented approach nor does it abstract the knowledge of the Mib's away from the system.
- The NTCIP mandatory and optional Mib's may expand with time as the standard continues to mature and meet new requirements. The solution for the NTCIP representation in CHART2 will need to be flexible enough that any change in the standard for supported Mib's doesn't require Java code addition.
- The CHART2 system is designed as a distributed and remotely available network, this requires that the protocol manage DMS's that are instance specific to the client manager. The Mib structure has to be available across the distributed network and wherever client managers may be. Any Mib solution has to be capable of being presented across the CORBA transport mechanism between client and server as required.
- The ASN.1 standard for presenting Mib's is a highly nodular manner of representing the Mib's but is poorly recorded in flat file format. Java does not have the parsing capabilities of C or Perl without the addition of outside libraries, and even then, those libraries are better served with regular expressions than straight parsing. The ASN.1 standard would need to be parsed into a format that is more Object Oriented and hence more Java suitable. The Mib structure is highly nodular as it is, the flat file representation of the ASN.1 standard is a poor and Java unfriendly way of displaying this nodularity. Technology for storing and displaying highly nodular data has improved with the addition of XML to the developers toolkit.
- The Mibs and mib variables contain more information than straight name:value pairs. If the Mib variables were to be hard coded into the classes then there would be a loss of precision in the data the classes would be able to present. The classes would need to know too much about the Mib variables as well, such as if they are readable

or writable. For instance a mib variable generally contains information such as, oid, system, object-type, syntax, status, description, access and size.

XML

The Extensible Markup Language is better known by it's three letter acronym, XML. It is an W3C approved standard as a meta-language for describing other languages. XML's benefit is that it does not specify the markup or the grammar for that language, allowing for great descriptive flexibility through a common set of tools and practices. This flexibility and extensibility require that XML have a large number of related standards for the handling of translation and data specification. XML is at it's most useful when describing unique system that require their data to be persistently stored in a highly nodular manner. A situation that is pertinent to the NTCIP Mib standard.

As XML is a popular language with open standards, there are many commercial and opensource libraries to increase productivity when developing applications requiring XML. The Apache XML project in particular includes many projects such as Xerces, Xalan, Cocoon and FOP which are extremely powerful and stable XM liraries. There are also other libraries such as Jdom, Jaxen, Dom4J, the latter which is an opesnource IBM project. Sun also includes in the J2EE distribution of the JDK many XML libraries. Edwards and Kelcey Technology has a great deal of experience with the Apache XML projects and with the Jdom project, having incorporated them into a previous projects with a great deal of success.

XML Solution

Using XML provides the best solution to the above problem domain. Using XML solves the problems inherent with the ASN.1 standard as it is recorded as a flat file format, and despite the tools for code generation of Mib names as class member variables, using XML for this purpose removes that aspect. It is important for the software code not to 'know' what it is requesting. As XML is fairly lightweight, the XML can be stored as XML in memory and manipulated in that manner. This gives lightweight database functionality in the software. Using XML to store and manipulate Mib's in this manner also allows for greater flexibility and expandability in describing the Mib's.

XML and XPath

XPath is the query language for XML, predominantly used in the stylesheet processing. XPath can also be used for the querying of XML tree's for specific information. With the Mibdb, the most common query will be for an OID and it's value. XPath is capable of traversing the tree and returning the node that satisfies the search criteria.

For example, to search the TS3.3 Mib which has been converted to XML;

```
/*
 * Get the Document from the Iterator
 */
Document document = (Document)it.next();
```

```
/*
 * Build the XPath expression
 */
XPath xpath = new XPath("//*[@" + MibDTD.ATTRIBUTE_TYPE + "='" + type + "']");

/*
 * Return the singular node.
 */
Element result = (Element)xpath.selectSingleNode(document);

/*
 * Create a clone of the element, and add to the OID object.
 */
oid = new OID((Element)result.clone());
```

The Document is the JDOM Document and represents an XML document in Java Objects. The XPath object is creating an XPath expression, in this case it is of the form;

```
"//*[@type='sysDescr']"
```

Where the MibDTD.ATTRIBUTE_TYPE is "type" and the type variable is "sysDescr" a System Group Mib. The resultant element is returned by applying the XPath criteria to the Document and this is then sent as an argument to the OID object which compositionally receives the element as a parameter for it's constructor. The Element is deep cloned before being passed to the OID. As a get only wants information, it does not need a reference to the original Document, just a copy of the data to present where it is requested.

XPath allows what would have been several recursive methods of searching the DOM nodes to be collapsed into two lines and with greater searching power and flexibility.

Mib XML DTD

The Mib structure has been fairly unchanging over the years, even though the data itself and the number of Mib points may have changed fairly consistently. As the XML is to represent the Mib's, the structure of the XML is best defined rigidly in a DTD or Document Type Definition. This is a template for the structure of the data. The Mib's are rigidly organized into Groups, Mib variables, Tables and Table Entries. Each of those major nodes has a series of attributes that need to represented in the DTD Schema.

```
<!--
      MIB DTD
      @author Cameron Riley
 -->

 <!ELEMENT root ANY>
 <!ELEMENT name (#PCDATA) EMPTY>
 <!ELEMENT definitions (#PCDATA) EMPTY>

 <!ELEMENT group (description?,mib*,entry*) EMPTY>
 <!ATTLIST group
   name CDATA #REQUIRED
   oid CDATA #REQUIRED
   system CDATA #REQUIRED
 >

 <!ELEMENT mib (description?) EMPTY>
 <!ATTLIST mib
   oid CDATA #REQUIRED
```

```
   system CDATA #REQUIRED
   type CDATA #REQUIRED
   syntax CDATA #REQUIRED
   size CDATA #IMPLIED
   access CDATA #REQUIRED
   status CDATA #REQUIRED
 >

 <!ELEMENT description (#PCDATA) EMPTY>

 <!ELEMENT entry (description?,mib*) EMPTY>
 <!ATTLIST entry
   oid CDATA #REQUIRED
   system CDATA #REQUIRED
   type CDATA #REQUIRED
   syntax CDATA #REQUIRED
   size CDATA #IMPLIED
   access CDATA #REQUIRED
   status CDATA #REQUIRED
   index CDATA #REQUIRED
 >
```

The Document Type Definition describes the root element as being 'root' which can have any tag as it's child. The element 'definitions' carries no other tags, only textual data. The 'group' tag which describes a Mib Group, has to have none or one 'description' tag and can have zero or many 'mib' tags. The 'group' tag has the mandatory attributes; @name, @oid and @system. The 'mib' tag has the mandatory attributes; @oid, @system, @type, @syntax, @access and @status. The 'mib' tag also has the implied attribute, @size. The 'mib' tag also has a 'description' child tag which can only appear once, or not at all.

Mib's represented as XML

Following the DTD above, the TS33 Mib with the Groups, System, SNMP and rs232 can be written in XML as below;

```
<?xml version="1.0"?>

<!--
      MIB TS3.3 Mibs.
      @author Cameron Riley

      MIB TS3.3
      MIB-XML created by Edwards and Kelcey Technology Inc. for the
      Maryland State Highways CHART2 project. The NTCIP standard
      specifies that the following groups from TS3.3 are mandatory;

       System Group
       SNMP Group
       rs232 Group
 -->



<!--
     Root Element
-->
<root>

  <!--
       Textual description of the MIB
  -->
  <name>
  TS33
```

```
</name>

<definitions>
  TS33bCLB.MIB
</definitions>



<!--
     The MIB Group.
-->
<group name="System" oid="1.3.6.1.2.1.1" system="1">
  <description>
    Implementation of the System group is mandatory for all
    systems.  If an agent is not configured to have a value
    for any of these variables, a string of length 0 is
    returned.
  </description>

  <mib oid="1.3.6.1.2.1.1.1.0"
       system="1.0"
       type="sysDescr"
       syntax="DisplayString"
       size="0-255"
       access="read-only"
       status="mandatory">
    <description>
      A textual description of the entity.  This value
      should include the full name and version
      identification of the system's hardware type,
      software operating-system, and networking
      software.  It is mandatory that this only contain
      printable ASCII characters.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.2.0"
       system="2.0"
       type="sysObjectID"
       syntax="OBJECT IDENTIFIER"
       access="read-only"
       status="mandatory">
    <description>
      The vendor's authoritative identification of the
      network management subsystem contained in the
      entity.  This value is allocated within the SMI
      enterprises subtree (1.3.6.1.4.1) and provides an
      easy and unambiguous means for determining `what
      kind of box' is being managed.  For example, if
      vendor `Flintstones, Inc.' was assigned the
      subtree 1.3.6.1.4.1.4242, it could assign the
      identifier 1.3.6.1.4.1.4242.1.1 to its `Fred
      Router'.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.3.0"
       system="3.0"
       type="sysUpTime"
       syntax="timeTicks"
       access="read-only"
       status="mandatory">
    <description>
      The time (in hundredths of a second) since the
      network management portion of the system was last
      re-initialized.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.4.0"
       system="4.0"
       type="sysContact"
       syntax="DisplayString"
       size="0..255"
```

```
       access="read-write"
       status="mandatory">
    <description>
      The textual identification of the contact person
      for this managed node, together with information
      on how to contact this person.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.5.0"
       system="5.0"
       type="sysName"
       syntax="DisplayString"
       size="0..255"
       access="read-write"
       status="mandatory">
    <description>
      An administratively-assigned name for this
      managed node.  By convention, this is the node's
      fully-qualified domain name.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.6.0"
       system="6.0"
       type="sysLocation"
       syntax="DisplayString"
       size="0..255"
       access="read-write"
       status="mandatory">
    <description>
      The physical location of this node (e.g.,
      'telephone closet, 3rd floor').
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.1.7.0"
       system="7.0"
       type="sysServices"
       syntax="INTEGER"
       size="0..127"
       access="read-only"
       status="mandatory">
    <description>
      A value which indicates the set of services that
      this entity primarily offers.

      The value is a sum.  This sum initially takes the
      value zero, Then, for each layer, L, in the range
      1 through 7, that this node performs transactions
      for, 2 raised to (L - 1) is added to the sum.  For
      example, a node which performs primarily routing
      functions would have a value of 4 (2^(3-1)).  In
      contrast, a node which is a host offering
      application services would have a value of 72
      (2^(4-1) + 2^(7-1)).  Note that in the context of
      the Internet suite of protocols, values should be
      calculated accordingly:

        layer  functionality
          1  physical (e.g., repeaters)
          2  datalink/subnetwork (e.g., bridges)
          3  internet (e.g., IP gateways)
          4  end-to-end  (e.g., IP hosts)
          7  applications (e.g., mail relays)

      For systems including OSI protocols, layers 5 and
      6 may also be counted.
    </description>
  </mib>

</group>


<!--
```

```
     The SNMP Group.

     Note : 1.3.6.1.2.1.11.7.0 is not used.
     Note : 1.3.6.1.2.1.11.23.0 is not used.
-->
<group name="SNMP" oid="1.3.6.1.2.1.11" system="11">
  <description>
    Implementation of the SNMP group is mandatory for all
    systems which support an SNMP protocol entity.  Some of
    the objects defined below will be zero-valued in those
    SNMP implementations that are optimized to support only
    those functions specific to either a management agent or
    a management station.  In particular, it should be
    observed that the objects below refer to an SNMP entity,
    and there may be several SNMP entities residing on a
    managed node (e.g., if the node is hosting acting as
    a management station).
  </description>

  <mib oid="1.3.6.1.2.1.11.1"
       system="1.0"
       type="snmpInPkts"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of Messages delivered to the
      SNMP entity from the transport service.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.2"
       system="2.0"
       type="snmpOutPkts"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Messages which were
      passed from the SNMP protocol entity to the
      transport service
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.3.0"
       system="3.0"
       type="snmpInBadVersions"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Messages which were
      delivered to the SNMP protocol entity and were for
      an unsupported SNMP version.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.4.0"
       system="4.0"
       type="snmpInBadCommunityNames"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Messages delivered to
      the SNMP protocol entity which used a SNMP
      community name not known to said entity.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.5.0"
       system="5.0"
       type="snmpInBadCommunityUses"
       syntax="Counter"
       access="read-only"
```

```
     status="mandatory">
  <description>
    The total number of SNMP Messages delivered to
    the SNMP protocol entity which represented an SNMP
    operation which was not allowed by the SNMP
    community named in the Message.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.6.0"
     system="6.0"
     type="snmpInASNParseErrors"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of ASN.1 or BER errors
    encountered by the SNMP protocol entity when
    decoding received SNMP Messages.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.8.0"
     system="8.0"
     type="snmpInTooBigs"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    delivered to the SNMP protocol entity and for
    which the value of the error-status field is
    'tooBig'.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.9.0"
     system="9.0"
     type="snmpInNoSuchNames"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    delivered to the SNMP protocol entity and for
    which the value of the error-status field is
    'noSuchName'.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.10.0"
     system="10.0"
     type="snmpInBadValues"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    delivered to the SNMP protocol entity and for
    which the value of the error-status field is
    'badValue'
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.11.0"
     system="11.0"
     type="snmpInReadOnlys"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number valid SNMP PDUs which were
    delivered to the SNMP protocol entity and for
    which the value of the error-status field is
    'readOnly'.  It should be noted that it is a
```

```
    protocol error to generate an SNMP PDU which
    contains the value `readOnly' in the error-status
    field, as such this object is provided as a means
    of detecting incorrect implementations of the
    SNMP.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.12.0"
     system="12.0"
     type="snmpInGenErrs"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    delivered to the SNMP protocol entity and for
    which the value of the error-status field is
    'genErr'
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.15.0"
     system="15.0"
     type="snmpInGetRequests"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP Get-Request PDUs which
    have been accepted and processed by the SNMP
    protocol entity
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.16.0"
     system="16.0"
     type="snmpInGetNexts"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP Get-Next PDUs which have
    been accepted and processed by the SNMP protocol
    entity
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.17.0"
     system="17.0"
     type="snmpInSetRequests"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP Set-Request PDUs which
    have been accepted and processed by the SNMP
    protocol entity.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.18.0"
     system="18.0"
     type="snmpInGetResponses"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP Get-Response PDUs which
    have been accepted and processed by the SNMP
    protocol entity
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.19.0"
```

```
     system="19.0"
     type="snmpInTraps"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP Trap PDUs which have
    been accepted and processed by the SNMP protocol
    entity.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.20.0"
     system="20.0"
     type="snmpOutTooBigs"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    generated by the SNMP protocol entity and for
    which the value of the error-status field is
    'tooBig'.
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.21.0"
     system="21.0"
     type="snmpOutNoSuchNames"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    generated by the SNMP protocol entity and for
    which the value of the error-status is
    'noSuchName'
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.22.0"
     system="22.0"
     type="snmpOutBadValues"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    generated by the SNMP protocol entity and for
    which the value of the error-status field is
    'badValue'
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.24.0"
     system="24.0"
     type="snmpOutGenErrors"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
    The total number of SNMP PDUs which were
    generated by the SNMP protocol entity and for
    which the value of the error-status field is
    'genErr'
  </description>
</mib>

<mib oid="1.3.6.1.2.1.11.25.0"
     system="25.0"
     type="snmpOutGetRequests"
     syntax="Counter"
     access="read-only"
     status="mandatory">
  <description>
```

```
      The total number of SNMP Get-Request PDUs which
      have been generated by the SNMP protocol entity.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.26.0"
       system="26.0"
       type="snmpOutGetNexts"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Get-Next PDUs which have
      been generated by the SNMP protocol entity
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.27.0"
       system="27.0"
       type="snmpOutSetRequests"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Set-Request PDUs which
      have been generated by the SNMP protocol entity.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.28.0"
       system="28.0"
       type="snmpOutGetResponses"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Get-Response PDUs which
      have been generated by the SNMP protocol entity.
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.29.0"
       system="29.0"
       type="snmpOutTraps"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      The total number of SNMP Trap PDUs which have
      been generated by the SNMP protocol entity
    </description>
  </mib>

  <mib oid="1.3.6.1.2.1.11.30.0"
       system="30.0"
       type="snmpEnableAuthenTraps"
       syntax="Counter"
       access="read-only"
       status="mandatory">
    <description>
      Indicates whether the SNMP agent process is
      permitted to generate authentication-failure
      traps.  The value of this object overrides any
      configuration information; as such, it provides a
      means whereby all authentication-failure traps may
      be disabled
    </description>
  </mib>

</group>

<!--
     The Generic RS232-Like Group.
-->
<group name="Generic RS232-Like" oid="1.3.6.1.2.1.10.33" system="33">
```

```
<description>
  Implementation of this group is mandatory for all
  systems that have RS-232-like hardware ports
  supporting higher level services such as character
  streams or network interfaces
</description>

<mib oid="1.3.6.1.2.1.10.33.1.0"
     system="1.0"
     type="rs232Number"
     syntax="INTEGER"
     access="read-only"
     status="mandatory">
  <description>
    The number of ports (regardless of their current
    state) in the RS-232-like general port table
  </description>
</mib>

<mib oid="1.3.6.1.2.1.10.33.2"
     system="2"
     type="rs232PortTable"
     syntax="sequence of Rs232PortEntry"
     access="not-accesible"
     status="mandatory">
  <description>
    A list of port entries.  The number of entries is
    given by the value of rs232Number
  </description>

  <entry oid="1.3.6.1.2.1.10.33.2.1"
         system="1"
         type="rs232PortEntry"
         syntax="Rs232PortEntry"
         access="not-accesible"
         status="mandatory"
         index="rs232PortIndex">
    <description>
      Status and parameter values for a port.
    </description>

    <mib oid="1.3.6.1.2.1.10.33.2.1.1"
         system="1.0"
         type="rs232PortIndex"
         syntax="INTEGER"
         access="read-only"
         status="mandatory">
      <description>
        A unique value for each port.  Its value ranges
        between 1 and the value of rs232Number.  By
        convention and if possible, hardware port numbers
        map directly to external connectors.  The value for
        each port must remain constant at least from one
        re-initialization of the network management agent to
        the next.
      </description>
    </mib>

    <mib oid="1.3.6.1.2.1.10.33.2.1.2"
         system="2.0"
         type="rs232PortType"
         syntax="INTEGER {other(1),rs232(2),rs422(3),rs423(4),v35(5)}"
         access="read-only"
         status="mandatory">
      <description>
        The port's hardware type
      </description>
    </mib>

    <mib oid="1.3.6.1.2.1.10.33.2.1.5"
         system="5.0"
         type="rs232PortInSpeed"
         syntax="INTEGER"
         access="read-write"
         status="mandatory">
```

```
          <description>
            The port's input speed in bits per second.
          </description>
        </mib>

        <mib oid="1.3.6.1.2.1.10.33.2.1.6"
             system="6.0"
             type="rs232PortOutSpeed"
             syntax="INTEGER"
             access="read-write"
             status="mandatory">
          <description>
            The port's output speed in bits per second.
          </description>
        </mib>

      </entry>

    </mib>


  </group>

</root>
```

CHART2 Mib XML Package High Level Design

For CHART2, the Mib package needs to be capable of four tasks, the first is a listing of the Mibs available, the second a listing of the Groups available in those Mib's. The Mib package must also be able to query a Mib variable for it's OID or value and also be able to set the value for an OID. As the OID is important to this functionality it is abstracted out into a separate object. The Mib Package consists of several objects, the MibPool, the Mibtable, the MibPoolFactory, the Mibdb, the OID, the MibManager interface, the OID exceptions and the MibService Facade.

The MibPool is the server based object which follows the Singleton pattern. It receives the Mib's abstracted into JDOM Documents and stores them as read-only in a table. The table is represented by a Mibtable, which is an object that extends the Hashtable class. The Mib's are loaded to the MibPool by the MibPoolFactory which converts flat file XML from either a filesystem, database or serialized object to JDOM Documents. The Mib's held in the tables of the MibPool are immutable. The MibPool offers one public method, the getPool method which returns a Mibdb. The getPool method deep clones it's tables and passes them into the Mibdb.

The Mibdb is the readable and writable DMS specific instance of the MibPool. The Mibdb carries the Mib's of the MibPool but in a client side instance which can be manipulated to store and change variables in the Mibdb. The Mibdb carries the methods to do many of the operations on it's own tables and follows the MibManager's public interface to achieve this.

The Facade for this package is the MibService object. The MibService presents a simple static series of methods to the CHART2 system, so objects within the CHART2 system have access to the more complex functionality behind the facade.

UML for Mib's as XML

The complete UML diagram for the Mib package is in an accompanying PDF file.

Diagram : MibPool and MibTable Object interaction

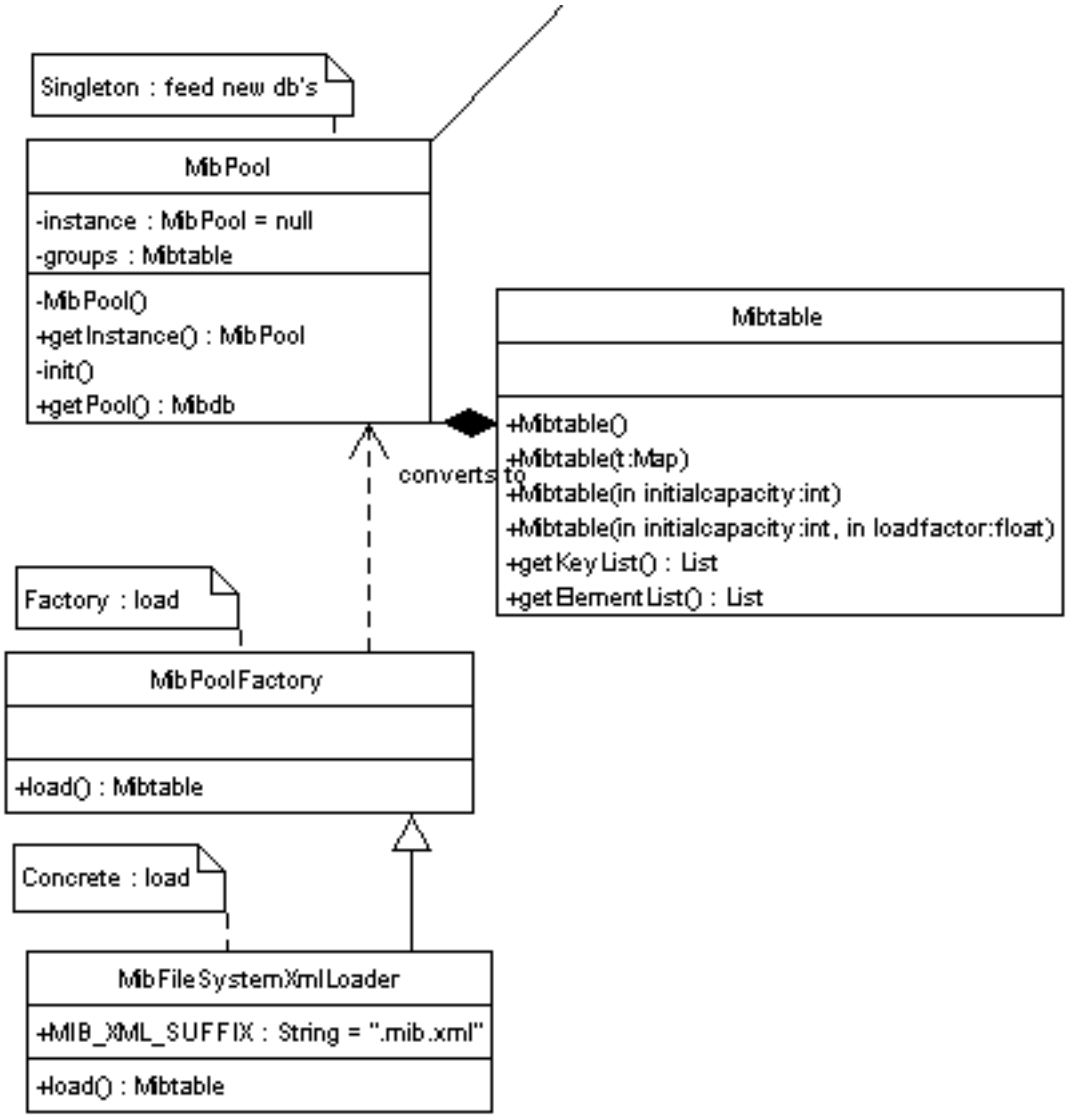
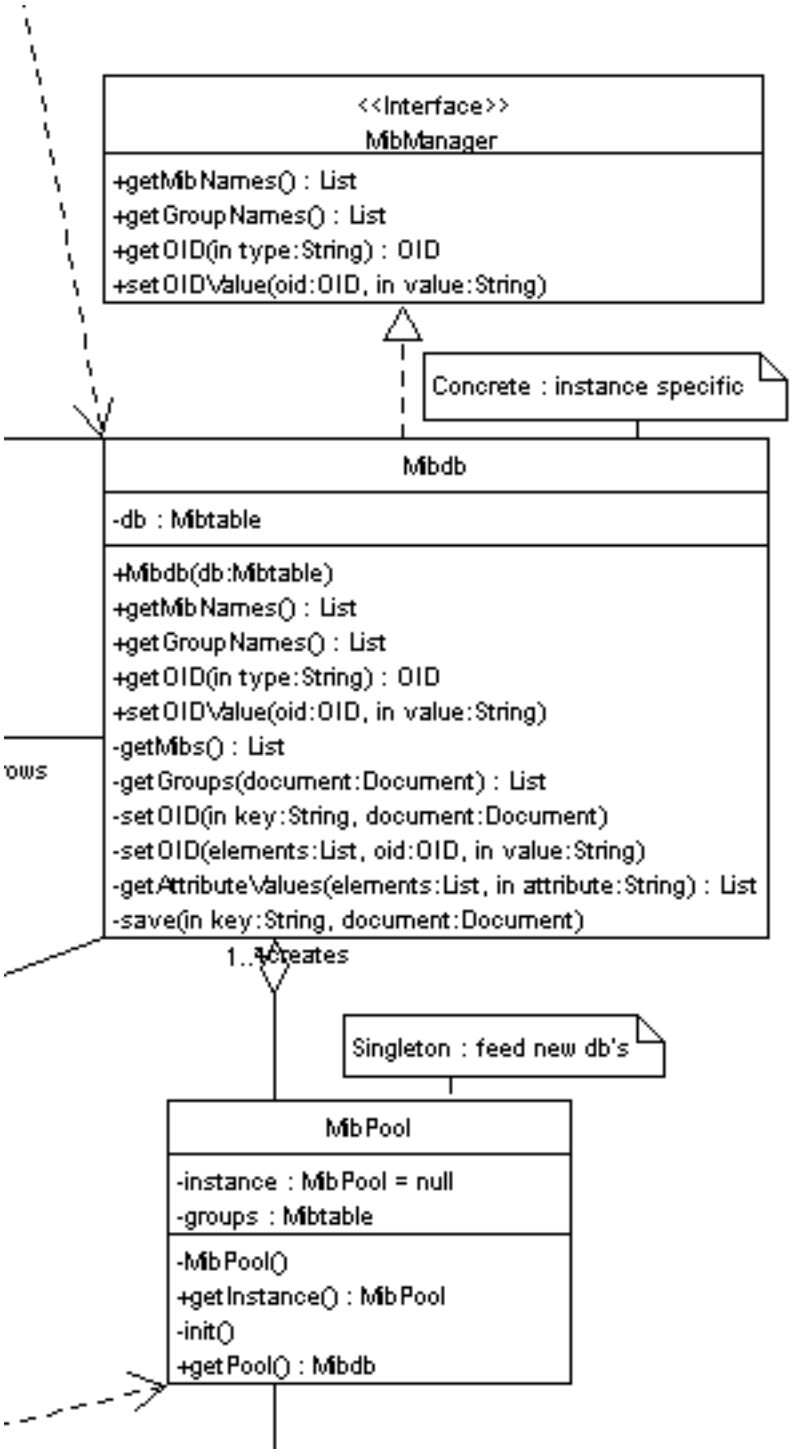


Diagram : MibService and MibManager Object interaction



Outside Libraries for Mib's as XML

Jdom is an opensource Java DOM library licensed under a variation of the BSD license. Jdom began as a project to ease Java XML development, the DOM model having been decided to unwieldy to use with the number of interfaces and factories. Jdom provides a lightweight XML set with Concrete classes for speed of development and ease of learning curve.

Jaxen is an opensource Java XPath Engine allowing for powerful XSL queries in Java of an existing DOM. Jaxen currently supports XPath for DOM,dom4j, EXML and Jdom. Jaxen is licensed under an Apache style license.

Xerces is an Apache project for Java XML Parser. The Xerces parser currently supports XML 1.0, XML Namespaces, DOM Level 2, Traversal and Range, SAX 2.0, JAXP 1.1 and XML Schema 1.0. Xerces is licensed under the Apache Public License.

CHART2 Mib Integration

The Mib package is at this point of time housed in the CHART2.Utility.Mib package and presents it's functionality and capabilities to the CHART2 system through the MibService facade. The MibService is the only manner in which any component of the CHART2 system should access the Mib functionality. The MibPool is currently only designed to load the NTCIP Mib's, further design and work would need to be done to make it flexible enough to discern between different DMS types' Mib's. The Mibdb is a generic Mib manipulator that cares not for what information it enacts on, only that the Mib's be in a table structure and stored as JDOM Documents in those tables. Currently the design representing Mib's as XML is a solution for NTCIP DMS's only.

Appendix 1 : The benefits of Opensource Libraries

Opensource projects and licenses are more prevalent today than they were 5 years ago, much of this is due to the media time Linux received during the technology boom of the 1990's. Opensource's value is it's code, it allows software projects to re-use existing, and more importantly tested code. An opensource project is defined by it's publicly published software code. The benefits are that the code is freely available for public use and more importantly for public contribution. This allows anyone to contribute to the project and advance the software codebase. This gives great empowerment to the developer and a large codebase of code to use in projects. With this comes the responsibility both legal and social to adhere to the license the code is published under. In some cases there are restrictions placed on how a derived work may be used, this is discussed further in the Appendix 2.

Edwards and Kelcey Technology has used opensource libraries in previous projects with great success. This has the benefit of our software engineering group not to have to re-invent the wheel each time to satisfy the requirements of a project. Several of our projects have used XML libraries, in particular the FMSuite series of products. As the focus of the FMSuite project was to solve operational problems with managing Facilities, by using the Apache XML libraries we were able to focus on the Facility rather than being sidetracked by having to solve low-level software engineering problems first. The other direct benefit is the capability of contributing to the project to advance it for our particular needs. This occurred with the Apache Turbine project.

Apart from the obvious saving's in cost by not purchasing an equivalent commercial library or software product, as an engineering company, the access to the source is empowering. It means we are not at the vendors beck and whim to add additional functionality. With an opensource project we are able to add the additional functionality as it is needed and then publish it back to the project for peer review.

Appendix 2 : Differences between Opensource Licenses

The main opensource licenses are the BSD license, the Apache Public License, the GNU Public License also known as the GPL, the Lesser GPL better known as the LGPL or the Library GPL and the Mozilla Public License commonly known as the MPL. All satisfy the opensource description defined by the opensource definition which can be found at www.opensource.org. There are also many other licenses which satisfy the opensource definition.

All licenses despite allowing for opensource software to be distributed have differences in what constitutes a derived work. This is where the licenses place restrictions on the manner in which the source can be distributed. A derived work is any body of work which incorporates the source code carrying one of these opensource licenses. There are different ways in which the source code can be incorporated into the new project, such as direct embedding, package integration, library integration or compile time linking.

The GPL has the most restrictive of what constitutes a derived work, given it's origins in a C world, static linking with a GPL'ed project is seen as being a derived work. Derived work also includes a project which uses any GPL licensed source code in it. If the work is not distributed and only used privately then none of this is an issue. It is only when the work is distributed, that to be distributable, the projects code to be distributed alongside GPL code, must be distributed under the GPL as well. This is an aspect of using GPL software that must be born in mind. If this is unacceptable to the

project, then GPL software cannot be used to create a derived work.

The LGPL allows for linking, importing and compiling against, to create a derived work without the projects source having to be licensed under the LGPL or GPL to be distributed. For this reason it is known as the Lesser or Library GPL. If however the project modifies the code in the LGPL'ed library and then distributes the library with the project, any modified code in the library must be published to whomever the project is distributed to. For most commercial concerns the LGPL is free enough and open enough without the GPL's restrictive clause.

The BSD license is the most liberal of licenses and used for the operating systems such as FreeBSD, OpenBSD and NetBSD. Other project which use the BSD license for their projects includes the X11 windowing system for Linux. The BSD license was also the basis for the Apache License. The BSD license states that the author of the code gives no warranty to it's function and that the authors copyright remain on the source. A BSD licensed project can be compiled to a binary without the derived work's source code having to be published to the customer or the public.

The Apache license is used for the well known Apache projects. Such as the Apache Webserver, Jakarta Tomcat, Jakarta Turbine, Xalan and Xerces XML projects. The Apache website is a wealthy resource for software engineers. The Apache license follows the BSD style of license.

The Mozilla Public License, or MPL grew out of the opensourcing of the Netscape codebase and was an attempt to balance opensource fundamentals with the wants and desires of for-profit business. A derived work with the MPL requires the modified code be published back to the project. The Mozilla Browser project is published under a dual MPL/GPL license and probably represents the most successful attempt at a commercial opensource project.

Appendix 3 : Edwards and Kelcey Commercial Support for Opensource Libraries

One of the recurring area's of concern for departments using opensource libraries or source code is the question of support. As opensource projects publish their code publicly, Edwards and Kelcey Technology can support any opensource source code used in any project. Edwards and Kelcey Technology can be contracted to do code maintenance on an opensource software project or add new functionality and features to the opensource software project. Edwards and Kelcey Technology has a great deal of experience supporting opensource libraries and projects.

Resources

- CHART2 : http://www.chart.state.md.us/
- Edwards and Kelcey Technology : http://www.ekcorp.com/
- Fop : http://xml.apache.org/fop/
- Jaxen : http://www.jaxen.org/
- Jdom : http://www.jdom.org/
- NEMA : http://www.nema.org/
- NTCIP : http://www.ntcip.org/
- Object Management Group : http://www.omg.org/
- Opensource : http://www.opensource.org/
- Xalan : http://xml.apache.org/xalan/
- Xerces : http://xml.apache.org/xerces/

References

- Edwards and Kelcey Report Subtask 1 : NTCIP Compliance Survey and Driver Development. 2001.
- Edwards and Kelcey Report Subtask 2 : NTCIP Driver High Level Design. 2001.
- Java and XML : Brett MacLaughlin. 2000.
- MSHA Report : CHARTII Release I, Build 2A - High Level Design.

Glossary

- AASHTO : American Association of State Highway and Transportation Officials.
- CORBA : Common Object Request Broker Architecture.
- DMS : Dynamic Message Sign.
- DTD : Document Type Definition.
- FMS : Field Management Station.
- HDLC : High-level Data Link Control.
- IDL : Interface Definition Language.
- ISDN : Integrated Services Digital Network.
- ITE : Institute of Transportation Engineers.
- MIB : Management Information Base.
- NEMA : National Electrical Manufacturers Association.

- NTCIP : National Transportation Communications for ITS Protocol.
- OID : Object Id.
- ORB : Object Request Broker.
- PDU : Protocol Data Unit.
- PMPP : Point to Multi Point Protocol.
- POA : Portable Object Adapter.
- SNMP : Simple Network Management Protocol.
- VMS : Variable Message Sign.
- W3C : World Wide Web Consortium.
- WAN : Wide Area Network.
- XML : Extensible Mark-up Language.